Algebra, Logic, Geometry: at the Foundation of CS

Tony Hoare
Honorary Professor at Griffith University

ICFEM 14 November 2018
Theses

• Foundations of the Theory of Programming can be taught as an aid to practical programming throughout a degree course in Computing Science.

• The Program Development Environment for all practical classes should be based on the Theory.

• The initial level of Math required is that of High School courses in Algebra, Logic, and Geometry.
Summary

1. Review of Boolean Algebra
2. Deductive Logic
3. Spatial and Temporal Logic
4. Unifying Theories of Programming
5. The Basic Principle of Concurrency
6. Unifying Theories of Concurrency
1. Review of Boolean Algebra

relevant also for
Hardware Design, Mathematics, and Philosophy.
George Boole (1815-1864)

• Professor of Mathematics at Queen’s College Cork, Ireland
• Book: 1854 An Investigation of the Laws of Thought proposed the binary algebraic operators not, and, or, and a binary comparison predicate: \( \leq \) (implication) as the foundation of a deductive logic of propositions \((p, q, r, \ldots )\)
Disjunction: $\lor$ (‘or’)  

- **Axioms:** $\lor$ is associative, commutative and idempotent  
- **Theorem:** $\lor$ distributes through $\lor$  
  \[(p \lor q) \lor r = (p \lor r) \lor (q \lor r)\]  
- **Proof:** $rhs = p \lor (r \lor (q \lor r))$ assoc  
  
  \[= p \lor ((q \lor r) \lor r) \quad \text{comm}\]  
  \[= p \lor (q \lor (r \lor r)) \quad \text{assoc}\]  
  \[= p \lor (q \lor r) \quad \text{idem}\]  
  \[= lhs \quad \text{assoc}\]  

Rightward distribution follows by comm
Geometry: \((p \lor q)\)

Venn diagram
Comparison: $p \leq r$

- Definition: $p \leq r$ as $r = p \lor r$
- $p$ is contained in $r$, $r$ contains $p$, $r$ is an expansion of $p$, ...

- Geometry:
V is an expansive operator

• Theorem: \( p \leq p \lor r \)

• Proof: \( p \lor r = p \lor p \lor r \) by idempotence

The theorem follows by definition of \( \leq \)

Here and henceforth we omit brackets around associative operators.
2. Deductive Logic

Relevant for all branches of mathematics, science and engineering
Aristotle 384-322 BC.

Founded the Academy in Athens, and lectured on

- sciences: physics, biology, zoology;
- aesthetics: poetry, theatre, music;
- ethics: politics, government, rhetoric;
- philosophy: metaphysics, logic, linguistics.

Recognised as the originator of logic and of classificatory biology, which uses his syllogisms for deducing consequences from its classifications.
Aristotle’s syllogism

\[
\begin{align*}
\text{All men are animals} & \quad \text{All animals are mortal} \\
\hline
\text{All men are mortal}
\end{align*}
\]

• If the two antecedents above the line have been proved the consequent below the line is also provable

• To validate a proof rule: first assume the antecedents and then use algebra prove the consequent

• To use a proof rule: first prove the antecedents and thereafter assume the consequent whenever required
Rule of Proof by Cases

\[ \frac{p \leq r \quad q \leq r}{(p \lor q) \leq r} \]

Validation: Assume the antecedents: \( r = p \lor r \) and \( r = q \lor r \)

\[
\begin{align*}
   r &= r \lor r & \text{the idempotence axiom} \\
   &= (p \lor r) \lor (q \lor r) & \text{by substitution for each } r \\
   &= (p \lor q) \lor r & \text{by distribution of } r \text{ through } \lor \\
\end{align*}
\]

The conclusion follows by the definition of \( \leq \)
Purpose of a proof rule

$$p \leq r \quad q \leq r$$

$$(p \lor q) \leq r$$

• To give a strategy for discovery of proofs:
  split a complex proof task (the consequent)
  into two or more simpler tasks (the antecedents)

• To reduce the overall length of the proof
Ordering: ≤

• Theorem: ≤ is a partial order
  
  reflexive: \( p \leq p \) by idempotence
  
  transitive: \[
  \frac{p \leq q \quad q \leq r}{p \leq r}
  \] by association
  
  antisymmetric: \[
  \frac{p \leq q \quad q \leq p}{p = q}
  \] by commutation
Covariance (monotonicity) of $\lor$

Theorem:

\[
\begin{align*}
    p \leq q & \quad \Rightarrow \quad p \lor r \leq q \lor r \\
    p \leq q & \quad \Rightarrow \quad r \lor p \leq r \lor q
\end{align*}
\]

Proof:

\[
\begin{align*}
p \leq q & \quad \leq q \lor r \quad \text{and} \quad r \leq q \lor r
\end{align*}
\]

The result follows by cases.

A covariant operator preserves the ordering of each of its operands

‘expanding a part can only expand the whole’

(and contracting similarly)
3. Spatial and Temporal Logic

for reasoning about what happens in space and time, e.g., inside a computer that executes a program
William of Occam (1287-1347)

• Franciscan friar, Scholar at Merton College Oxford
• excommunicated (1328) rehabilitated (1359)
• Occam’s razor: entities should not be postulated beyond necessity
• Book: Summa Logicae (1323)

with operators for implication, disjunction, conjunction, causation; and temporal operators while (|), and then (;)
Space and Time

- \( p, q, r, \ldots \) are propositions that describe actions occurring inside some region of space during some interval of time
- e.g., A History of France 1789-1815

- A program describes any execution the program of it on any computer system and on any occasion.
- The actions are executions of the basic commands of the program.
Sequential and concurrent composition

- An execution of a composition (\( p; q \) or \( p \mid q \)) performs all actions of \( p \) and also of \( q \) where there is no action common to both executions.
- \( p; q \) starts simultaneously with the start of \( p \), and ends with the end of \( q \); furthermore, \( q \) starts only when \( p \) ends.
- \( p \mid q \) starts with the start both \( p \) and \( q \), and ends with the end of both of them \( q \). Its duration is the maximum of their durations.
Algebraic Axioms

• [ ] describes an interval in which nothing happens (aka null, skip)
• ; is associative and has unit [ ]
• ; distributes through $\lor$ (both leftward and rightward):
  
  e.g., $p;(q \lor q') = (p;q \lor p;q')$

Explanation: both sides start with $p$ and end with either $q$ or $q'$. In each case, the chosen alternative starts when $p$ ends.

Distribution justifies giving ; a precedence stronger than $\lor$
Sequential proof by cases

• Theorem: \[ \frac{p; q \leq r}{(p \lor p'); q \leq r} \]

• Assume (1) \( r = p; q \lor r \) and (2) \( r = p'; q \lor r \)

Therefore \( r = p; q \lor p'; q \lor r \) substitute (2) in (1) 

\[ = (p \lor p'); q \lor r \] ; distributes thru \( \lor \)

We have proved the consequent of the rule (by definition of \( \leq \))
Proof rule for sequential composition

• Theorem: \[
\begin{align*}
& \quad p; q \leq m \\
\hline
& \quad m; r \leq t \\
& \quad p; q; r \leq t
\end{align*}
\]

• Assume (1) \( m = p; q \lor m \) and (2) \( t = m; r \lor t \)

Therefore \[ t = (p; q \lor m); r \lor t \]
\[ = p; q; r \lor m; r \lor t \]
\[ = p; q; r \lor t \]

substitute (1) in (2)
distribute \( r \) thru \( \lor \)
substitute back by (2)
Corollaries

• Theorem:

\[
\begin{align*}
\text{If } p; q &\leq m & \text{then } m; r &\leq t \\
\hline
\text{then } p; q; r &\leq t
\end{align*}
\]

\[
\begin{align*}
\text{If } p &\leq m \\
\hline
\text{then } m; r &\leq t \\
\text{then } p; r &\leq t
\end{align*}
\]

\[
\begin{align*}
\text{If } p; q &\leq m \\
\hline
\text{then } m &\leq t \\
\text{then } p; q &\leq t
\end{align*}
\]

Proof: by substitution of \([\ ]\) for \(q\) and in the first, and for \(r\) in the second
4. Unifying Theories of Programming

Hoare’s axioms for verification of programs are equivalent to Milner’s operational rules, describing abstractly the implementation of the same programming language.
Alternative Interpretations of $p;q \leq r$

1. If $p$ describes what happens from the start of $r$, and $q$ describes what happens from the end of $p$ up to the end of $r$, then $r$ correctly describes the whole of $(p;q)$

2. To execute $r$ an implementation may first execute $p$, followed by $q$
Hoare Triples

• Define \( \{p\} q \{r\} \) as \( p;q \leq r \)
• Substitution of this definition in the Proof Rule for \( ; \) gives

\[
\begin{array}{c}
\frac{\{p\} q \{m\} \quad \{m\} r \{t\}}{\{p\} q;r \{t\}}
\end{array}
\]

which is the Hoare rule for \( ; \)

• The two corollaries give:

\[
\begin{array}{c}
\frac{\{p\} q \{m\} \quad m \leq t}{\{p\} q \{t\}} \quad \frac{p \leq m \quad \{m\} r \{t\}}{\{p\} r \{t\}}
\end{array}
\]

which are the Hoare rules of Consequence
Milner Transitions

- Define $r \rightarrow q p$ as $q;p \leq r$.  
  Note the change of order
- Substitution in the Proof Rule for $; \rightarrow$ now gives

\[
\frac{m \rightarrow q p \quad t \rightarrow r m}{t \rightarrow q;r m \rightarrow p}
\]

the Milner rule for $; \rightarrow$
- The two corollaries are

\[
\frac{t \leq m \quad m \rightarrow q p}{t \rightarrow q p}
\]

\[
\frac{t \rightarrow r m \quad m \leq p}{t \rightarrow r p}
\]

i.e., Milner’s ‘structural rules’, with $\equiv$ replaced by $\leq$
Applications to Programming

• Most interpreters and compilers for programming languages follow an operational semantics expressed as Milner Transitions.
• Most program analysers and proof tools for sequential languages follow a verification semantics expressed Hoare Triples.
• Many papers and theses in the Theory of Programming prove the consistency between these two ‘rival’ Theories.
• Algebra unifies them by proofs which could be found or understood by a first-year university student.
5. The Basic Principle of Concurrency

| has the same laws as; the interaction between them is expressed in an Interchange law, which permits a concurrent program to be executed sequentially by interleaving.
Algebraic Axioms for $|$ 

- $|$ is associative with unit $[ ]$
- $|$ distributes through $\lor$
- $(p \mid q);(p' \mid q') \leq (p;p') \mid (q;q')$ (interchange) 
  interchanges the operator $;$ with $\mid$, and operand $p'$ with $q$
- $|$ is not necessarily commutative (e.g., the UNIX pipe)
Interchange

\[(p \| q);(p' \| q') \leq (p;p') \| (q;q')\]

Theorems: \(p;q' \leq p \| q'\) by interchange, with \(p' = q = []\)

\(q;p' \leq p' \| q\) similarly, with \(q' = p = []\)

\(p;q \lor q;p \leq p \| q\) by the rule for cases

The concurrency on the right can be implemented by non-deterministic choice of either of the two interleavings of its two operands on the left. Generalisation to any number of concurrent compositions proves the Basic Principle of Concurrency: a concurrent program can be run on a computer with any lesser number of processors.
\[(p \mid q);(p' \mid q') \leq (p;p') \mid (q;q')\]

Theorems

\[(p \mid q);q' \leq p \mid (q;q')\]
\[p;(p' \mid q') \leq (p;p') \mid q'\]
\[q;(p' \mid q') \leq p' \mid (q;q')\]
\[(p \mid q);p' \leq (p;p') \mid q\]

\[p' = []\]
\[q = []\]
\[q' = []\]

All four are proved by substitution of [ ]

They are known as small interchange laws. They are used for the first step in deriving the four interleavings of terms with three operands.
Example of Interleaving

\[ abcd \mid xyzw \quad rhs \text{ of interchange} \]
\[ \geq (a;bcd) \mid (xy;zw) \quad \text{associativity (twice)} \]
\[ \geq (a \mid xy);(bcd \mid zw) \quad \text{interchange} \]
\[ \geq (a \mid x;y);(b;cd \mid zw) \quad \text{associativity (twice)} \]
\[ \geq (a \mid x);y;(b \mid zw);cd \quad \text{small interchange (twice)} \]

\[
\begin{array}{c}
\vld \vld \vld \vld \\
\vld \vld \vld \\
\vld \vld \\
\vld \vld \\
\end{array}
\]

\[ xayzbwcd \quad \text{similarly} \]

Each step of the proof reduces the length of a red or black substring. So termination is assured.
Completeness of Interchange

• Metatheorem (Fundamental Principle of Concurrent Programming):
Let $x$ be any interleaving of terms $y$ with $z$ (with colours preserved). Then $x \leq y \mid z$ can be proved from the axioms given so far.

Proof: Repeatedly apply the interchange law to $x$ (from left to right), subject to the constraint that $p$ and $p'$ are strings wholly from $y$, and $q$ and $q'$ are strings wholly from $z$. On each application, at least one substring of the same colour will get longer, and the end result will be the term $y \mid z$ itself.
6. Unifying Theories of Concurrency

The unification of Hoare and Milner Theories is extended to concurrency.
Interchange Rule

\[
\frac{p; q \leq r}{p'; q' \leq r'}
\]

\[(p \mid p') ; (q \mid q') \leq (r \mid r')\]

The rule tells how to prove a complicated concurrent theorem by splitting it into two proofs of two much simpler sequential theorems.

Theorem: This rule is equivalent to the Interchange Law

Proof: next two slides
The axiom implies the rule

Assume the antecedents of the rule: \( p; q \leq r' \) and \( p'; q' \leq r' \)

\[
(p; q) \mid (p'; q') \leq (r \mid r') \quad \text{(covariance twice)}
\]

\[
(p \mid p'); (q \mid q') \leq (p; q) \mid (p'; q') \quad \text{(interchange law)}
\]

So \( (p \mid p'); (q \mid q') \leq (r \mid r') \quad \text{(by transitivity of \( \leq \))}

Therefore

\[
\frac{p; q \leq r}{(p \mid p'); (q \mid q') \leq (r \mid r')} \quad \frac{p'; q' \leq r'}{(p \mid p'); (q \mid q') \leq (r \mid r')} \quad \text{(the modularity rule)}
\]
The rule implies axiom

\[ p; q \leq r \quad \frac{p'; q' \leq r'}{(p \mid p')(q \mid q') \leq (r \mid r')} \] (concurrency rule)

Proof: Replace \( r \) by \( p; q \) and \( r' \) by \( p'; q' \) throughout.

The antecedents true by the reflexivity of \( \leq \)
and the conclusion is:

\[ (p \mid p')(q \mid q') \leq (p; q) \mid (p'; q') \]
which is the interchange law.
\[
\begin{align*}
\{\mathit{p}\} \quad \{\mathit{q}\} \quad \{\mathit{r}\} & \quad \{\mathit{p}'\} \quad \{\mathit{q}'\} \quad \{\mathit{r}'\} \\
\{\mathit{p} \parallel \mathit{p}'\} \quad \{\mathit{q} \parallel \mathit{q}'\} \quad \{\mathit{r} \parallel \mathit{r}'\} & \\
\end{align*}
\]

Separation Logic (O’Hearn)

\[
\begin{align*}
\mathit{r} \xrightarrow{\mathit{q}} \mathit{p} & \quad \mathit{r}' \xrightarrow{\mathit{q}'} \mathit{p}' \\
\rightarrow & \\
(\mathit{r} \parallel \mathit{r}') & \xrightarrow{(\mathit{q} \parallel \mathit{q}')} (\mathit{p} \parallel \mathit{p}') \\
\end{align*}
\]

Operational Semantics (Milner’s CCS)

\[
\begin{align*}
\mathit{p};\mathit{q} \leq \mathit{r} & \quad \mathit{p}';\mathit{q}' \leq \mathit{r}' \\
\rightarrow & \\
(\mathit{p} \parallel \mathit{p}');(\mathit{q} \parallel \mathit{q}') & \leq (\mathit{r} \parallel \mathit{r}') \\
\end{align*}
\]

concurrency rule

are all equivalent to the Interchange Law
7. Conclusion

Merits and Deficiencies of Algebra
In Praise of Algebra

• Simple, elegant, reusable, tractable by people and by machines,
• Algebraic transformation is essential in the top-down design of application system architecture by successive refinement
• Directly used in compilation, optimisation, refactorization, obfuscation, and automatic generation of program code
• Algebra unifies theories which underlie a range of programming tools, and is essential for their correct interworking...
• and for an effective education in Computer Science
Deficiencies

• It has insufficient expressive power: no quantification.
• Logic can also specify and verify interfaces between components of a program.
• It cannot specify basic commands.
• Logic specifies basic commands (assignment, input, output, ...)
• It has no negation: it cannot prove that a formula is not a theorem.
• Geometry is a model of both algebra and logic. It provides test cases for incorrect programs and counterexamples for false conjectures.

• see  https://www.cl.cam.ac.uk/~carh4/
Anybody against Algebra?
Isaac Newton (1642-1726)

Communication with Richard Gregory (1694)

“Our [my] specious [falsely convincing] algebra [the infinitesimal calculus] is fit enough to find out [has some heuristic value], but entirely unfit to consign to writing and commit to posterity [it cannot and must not be published].”

(with translation to Modern English)

Newton’s proofs were geometric, establishing properties of the Keplerian ellipses that describe the orbits of the planets
Bertrand Russell (1872 – 1970)

The method of “postulating” what we want has many advantages; they are the same as the advantages of theft over honest toil. Let us leave them to others and proceed with our honest toil.

Introduction to Mathematical Philosophy.

Russell then refused to postulate the existence of real numbers (such as the sqrt of 2), and proceeded to model them by the Dedekind cut.
Gottfried Leibniz (1646-1716)

• calculemus

Let us calculate (symbolically)
Acknowledgements to my Colleagues

• at Oxford University: Jean-Raymond Abrial, Richard Bird, Joe Stoy, Dana Scott, Bill Roscoe, Steve Brookes, Carroll Morgan
• at Microsoft Research: Jade Alglave, Ernie Cohen, Byron Cook, Rustan Leino, Matt Parkinson, Jon Pincus
• at Cambridge and Griffith University: Nada Amin, Zhe Hou
• my recent coauthors: Bernhard Moeller, Peter O’Hearn, Stefan van Staden, Georg Struth, Ian Wehrman, John Wickerson
• my heroes: Bertrand Russell, Edsger W. Dijkstra, Robin Milner